THE FORMAL SPECIFICATION OF THE DOCUMENT STRUCTURES OF THE ODA STANDARD

WOLFGANG APPELT* RICHARD CARR** GERNOT RICHTER*

* *Gesellschaft für Mathematik und Datenverarbeitung, Sankt Augustin, FRG*

** *National Computing Centre, Manchester, UK*

Abstract:
The Office Document Architecture (ODA) is an International Standard, currently developed by TC 97/ SC 18 of the International Organization for Standardization (ISO). This paper describes the current state of a formal specification of the ODA document structures by mathematical means.

1. Introduction

For several years major efforts concerning the development of standards in the field of document processing have taken place within the International Organization for Standardization (ISO). As a result of this work an "*Office Document Architecture (ODA) and Interchange Format*" was developed which will become an International Standard in the near future [ISO8613]. The purpose of the standard is to facilitate the interchange of office documents, such as reports, invoices, letters etc., where the interchange is by electronic means, e.g., by electronic mail. The standard is divided into several parts, including "*Document structures*", "*Document processing reference model*", "*Office Document interchange format*", "*Character content architecture*" and "*Geometric graphics architecture*", consisting of several hundred pages of text. As it is usually the case with such standards, the means of expression by which the standard is defined, is natural English. The publication of the standard is expected for 1988.

In spring 1985 a special working group within ISO started to develop a formal specification of the ODA Standard [Appelt, Carr, Richter, 87]. The work was concentrated mainly on part 2 of [ISO8613], called "*Document structures*", since this part plays a central role: The *document structures* can be regarded as the skeleton of the whole standard, supporting the other parts, e.g., the different content types of a document as *character contents*,

geometric graphics or *raster graphics*. Furthermore, because of the complexity of this part of the standard, a formal specification was considered to be very useful. It is expected that after the usual balloting procedures among the ISO member bodies the formal specification will become an addendum to the ODA Standard.

2. Aims of the formal specification

The formal specification of the ODA Standard has been developed with the aims of providing

— an unambiguous and consistent interpretation of the standard,
— a reference point for examining effects of revisions to the standard,
— a basis for implementations of the standard and
— a basis for the verification of systems conforming to the standard.

One of the major advantages of applying a formal specification technique in addition to a natural language text is that it provides a single interpretation of that text during the specification, implementation and utilisation process. It avoids different views of its information structures, functions and processes and hence different interpretations of the natural language text.

The formal specification can also be used as a basis for further developments since the specification can provide an ideal model for accurately and economically predicting the effects of changes and extensions to the standard.

A further advantage of formalising the standard is that potential implementors will be provided with a more precise specification with which they can work. By considering the formal specification rather than relying entirely on the natural language text readers can reason more thoroughly and gain better insights into the standard. This will greatly reduce the risk of producing non–conforming implementations.

Testing conformance to the standard involves the development of test tools that determine whether a system can correctly transmit and receive conforming data streams. In the development of such tools, the formal specification is expected to ensure that the testing software is evaluating the system against the valid interpretation of the standard.

A side–effect of the work on the formal specification was the detection of a number of inconsistencies and ambiguities in the natural language text. Corresponding comments were produced and submitted to the working group responsible for the development of the natural English version of the ODA Standard.

3. Method of the formal specification

An appropriate method for a formal specification of ODA document structures should have several, partially conflicting properties:

— To provide a rigorous and unambiguous specification the method must rely on a formal language, based on well defined syntactical and semantical constructs.

— Such a language should be especially appropriate for the constructs and cross relationships found within ODA documents.

— The language should be easy to learn; it should not prerequisite a specific knowledge, that cannot be expected from the average reader/implementor of the ODA Standard.

— The formal specification should be as close as possible to the natural language standard; relationships between specific clauses of the natural language text and its formal counterpart should be obvious.

The formal approach that was chosen after reviewing several alternatives is based on an information model developed by *Durchholz* and *Richter*, called "*Information Modelling by Composition*" (IMC) [Richter, Durchholz, 88]. An associated formal language, called *IMC–language* or *IMCL*, whose principles will be illustrated in the next section, is also available. (An earlier version is applied in [Richter, Durchholz, 82].) This language has its mathematical foundations in elementary set theory and first–order predicate logic. The relevant part of the language is described in a short self–contained annex to the formal specification of the Standard and a reader with an education in computer sciences or mathematics should be able to understand the language and following this the formal specification itself. One of the main reasons for choosing compositional information modelling was that it meets in particular the second and the fourth of the above mentioned requirements and, not less than comparable formalisms, the third one. The first criterion is exspected to be met by any sound formalism.

Using IMCL the complete formal specification of ODA document structures yields about 150 so–called *definitions* (first–order formulae), where each definition usually describes a specific property or relation in a rigorous, formal way (see section 6). Furthermore, each definition has an associated semi–formal English explanation. Though these semi–formal English text clauses do not belong to the formal specification, they provide links to the natural language text of the ODA Standard by referencing explicitly those clauses in the standard that are under definition. This gives some indirect confirmation about the completeness of the specification. Furthermore, the formal specification is structured to provide a high level of decomposition

which aids the implementor by revealing just those parts of the standard under consideration at any time.

4. Survey of IMCL

Though a complete description of IMCL is not possible within this paper a short survey of the basic principles shall be given within this section.
The "world" of IMCL consists of

— the *universe* which in turn consists of abstract elements called *entities*; an *entity* may be a *construct*, a *spot* or a *spotset* (i.e., a set of *spots*),
— *predicates* on *entities* of the *universe* and
— *operators* on *entities* of the *universe.*

A *construct* is either

— an *atomic construct* or
— a *composite construct* (a *composite construct* may be a *collection*, a *catenation* or a *nomination*).

A *collection* is a set of constructs in the sense of mathematical set theory. A *catenation* can be regarded as a sequence of constructs where each construct has a preceding construct (except the first one) and a succeeding construct (except the last one). A *nomination* can be regarded as a set of pairs (*name*,*construct*) where each *name* (usually an atomic construct) is unique within the *nomination.* As a consequence, within a *nomination* each element can be addressed by its *name.*

Atomic constructs are either numbers or atoms without built–in semantics. They are syntactically denoted as character strings, e.g.,

```
314   'a'   'data'
```

Note the single quotes enclosing a character or character string to denote an *atomic construct* without built–in semantics. *Composite constructs* are denoted by brackets and their component terms are separated by semicolons (in the case of *collections* or *nominations*) or by arrows (in the case of *catenations*), e.g.,

```
[314; 'a'; 'data'] [→'d'→'a'→'t'→'a'→] ['pi':314; 'std':'ODA']
```

The first expression denotes the collection of the three atomic constructs 314, 'a' and 'data'. The second expression denotes a catenation consisting of the atomic constructs 'd', 'a' and 't'. Notice the difference between the atomic construct 'data' and the composite construct [→'d'→'a'→'t'→'a'→]. The third expression denotes a nomination with two components, one having the name 'pi' with the associated component construct 314, the other one having the name 'std' with the associated component construct 'ODA'. A name and its associated component construct are syntactically separated by

a colon. The empty collection, catenation or nomination is denoted by [], [→] or [:], respectively. Of course, more complicated composite constructs may be formed by nesting collections, catenations and nominations, e.g.,

[314; [→'d'→'a'→'t'→'a'→]; ['pi':314; 'std':[→'O'→'D'→'A'→]]]

Using collections, catenations and nominations, rather complex structures and especially many of the structures which are encountered in the modelling of information in computer sciences can be built–up. To allow the decomposition of composite constructs or to address a specific construct within a composite construct the concept of a *spot* has been introduced. For example, the component constructs of the catenation [→'d'→'a'→'t'→'a'→] are 'a', 'd' and 't'. Whereas the atomic constructs 'd' and 't' appear at one *spot* only, the atomic construct 'a' appears at two *spots*, namely at the second and at the fourth position counted from the front end. So, [→'d'→'a'→'t'→'a'→] has four component *spots*, but only three component *constructs*. If a construct is considered outside of any context, it is said to be at its *ownspot*.

At a first glance, constructs might appear to play the central role in information modelling. However, it is the spots that are the primary objects of manipulation and therefore they are often more prominent in actual applications of information handling than the constructs themselves. Usually they are selected by selection criteria. A selection criterion need not give a single spot: in many cases applying a selection criterion on some construct will return a set of spots, called a *spotset*. Thus, the objects most naturally dealt with are not even spots, but rather *spotsets*. So in IMCL, the emphasis is on *spotsets* rather than on spots. If a spotset contains only one spot, it is called a *singleton spotset*.

Several unary and binary *predicates* with built–in semantics are defined in IMCL. A *predicate* is either True or False. For example,

$t_1 = t_2$	is True, iff (if and only if) the entities t_1 and t_2 are identical;
$t_1 \in t_2$	is True, iff the entity t_1 is a member of the collection t_2;
$t_1 \hat{\in} t_2$	is True, iff the entity t_1 is a singleton spotset and a subset of the spotset t_2;
$t_1 \subset t_2$	is True, iff t_1 is a subset of t_2 and t_1 and t_2 are both collections or spotsets, respectively;
IsAtom(t)	is True, iff t is an atomic construct;
IsNumber(t)	is True, iff t is a number;
IsCol(t)	is True, iff t is a collection;
IsCat(t)	is True, iff t is a catenation;
IsNom(t)	is True, iff t is a nomination;

IsSpotset(t) is True, iff t is a spotset;
IsSingle(t) is True, iff t is a singleton spotset.

IMCL also includes several *operators* with built–in semantics. For example, for entities being numbers the usual arithmetical operations $+ \; - \; * \; /$ are defined. For collections or spotsets the usual set theoretical operations $\cup \; \cap$ \ (union, intersection, set difference) are defined. Some additional IMCL operators are, for example:

$\hat{}t$ If t denotes a construct, $\hat{}t$ denotes the singleton spotset whose element is the ownspot of the construct given by t (read "ownspot of").

$t\bullet$ If t denotes a spotset without atomic spots, $t\bullet$ denotes the set of all spots which are immediately inward the spots given by t (read "next inward").

C t If t denotes a singleton spotset, C t denotes the component construct at the spot given by t.

N t If t denotes a set of exactly one spot immediately inward a nomination spot, N t denotes the name at the spot given by t.

CARD t If t denotes a collection or a spotset, CARD t returns its cardinality, i.e., the number of component constructs or spots within it.

t FIRST If t denotes a spotset whose spots are immediate component spots of catenations, t FIRST denotes the set of those spots of t whose position from the front end is 1.

Since IMCL is a first–order language based on predicate logic the usual logical quantifiers $\forall$ (for all) and $\exists$ (exists) are part of the language as well as the logical connections *not*, *and* , *or* , *iff* and *impl* (implies). For successive quantification the usual notational simplification is used, i.e.,

$\exists x(x \in \text{m}$ *and* formula) may be abbreviated as: $\exists x \in \text{m}$ (formula)
$\forall x(x \in \text{m}$ *impl* formula) may be abbreviated as: $\forall x \in \text{m}$ (formula)

Some simple examples may illustrate the use of the predicates and operators. For t being the construct

[[→'d'→'a'→'t'→'a'→]; 'ODA']

IsCol(t) is True whereas IsAtom(t) or IsCat(t) are False. CARD t returns the number 2 since t contains the two constructs [→'d'→'a'→'t'→'a'→] (a catenation) and 'ODA' (an atomic construct).

$\hat{}t$ denotes the singleton spotset containing as the only element the ownspot of [[→'d'→'a'→'t'→'a'→]; 'ODA']. Consequently, $\hat{}t\bullet$ denotes the spotset containing the two component spots with the constructs [→'d'→'a'→'t'→'a'→] and 'ODA'. Starting from the ownspot of [→'d'→'a'→'t'→'a'→] the expression

$$\hat{}[\rightarrow\text{'d'}\rightarrow\text{'a'}\rightarrow\text{'t'}\rightarrow\text{'a'}\rightarrow] \bullet \text{FIRST}$$

returns the singleton spotset with the spot of 'd', and the expression

$$\text{CARD}\ (\hat{}[\rightarrow\text{'d'}\rightarrow\text{'a'}\rightarrow\text{'t'}\rightarrow\text{'a'}\rightarrow]) \bullet$$

returns the number 4 (four component spots).

5. Document models in ODA and their specification

ODA provides two kinds of abstraction in its architecture. The *structural model* is used to model the logical and layout objects found in a document while the *descriptive representation* provides a collection of data structures, called *constituents*, which hold information on these objects, as shown in figure 1.

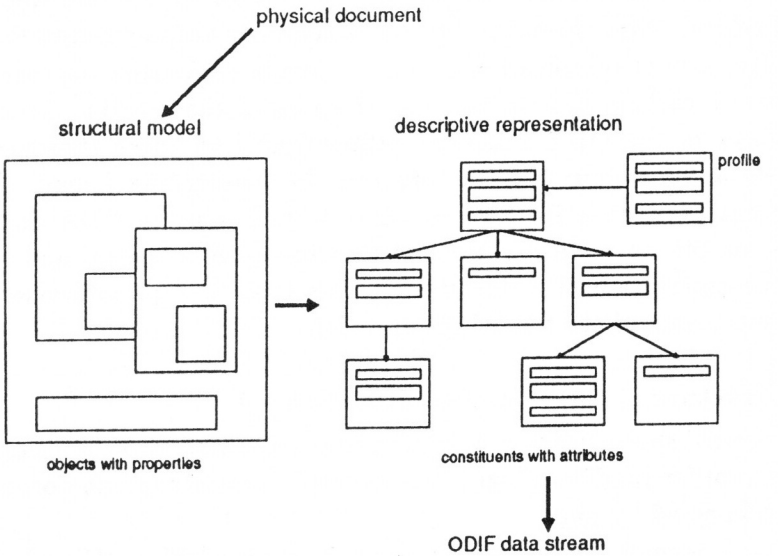

Figure 1: Structural model and descriptive representation in ODA

The abbreviation *ODIF* stands for "Office Document Interchange Format" which is described in part 5 of [ISO8613]. ODIF defines the binary coding format of the descriptive representation of an ODA document and is based on the Abstract Syntax Notation One (ASN.1). ODA does not require that an implementation uses the ODIF format for the internal representation of its documents; other formats might be more suitable for a specific implementation. For interchange purposes, however, ODIF is the format proposed in the standard.

The objects in the *structural model* are represented in the *descriptive representation* by *constituents* with *attributes*. That is, in the descriptive

representation, a *constituent* is a *set of attributes* used to describe an object and its properties in the structural model. For example, a *constituent* 'page' has attributes describing its *dimensions*, the *content type* it contains (e.g., character contents or raster graphics), the *subordinate objects* it has, etc.

The full descriptive representation of an ODA document is a *set of constituents* as shown in figure 2 on the next page. It contains a constituent called the *document profile* together with the constituents that form the *document body*. The body, in turn, comprises the subsets of constituents called the *generic part*, the *specific part* and the *style part*.

The *formal specification* of the ODA document structures is concerned with the *descriptive representations* of documents since constituents and their attributes are the basis of the interchange stream and their make–up and relationships the subject of ODA conformance requirements.

The formal specification pursues a *purely declarative approach* to information structure description, i.e., it specifies the possible structures that conform to the ODA Standard rather than how these structures are obtained as an output of some process. In the current formal specification this has been done for all structures occurring in an ODA document except for the interrelationships between the specific logical and the specific layout description of a document. This topic will be addressed in future enhancements of the formal specification.

6. Structure of the formal specification

The formal specification of the ODA document structures is a single formula in first-order predicate logic. The formula consists of sub–formulae which are connected by *and*:

$$\text{formula}_1 \ \underline{\textit{and}} \ \text{formula}_2 \ \underline{\textit{and}} \ \text{formula}_3 \ \underline{\textit{and}} \ \ldots \ \text{formula}_n$$

A formula is also called a *definition* since it either defines a *concept* used in the natural English description of the ODA Standard, or a so–called *subsidiary predicate* or *subsidiary function* which has been introduced for the sake of readability.

To make the formal specification more readable and to provide an easy link to the natural English text the definitions are presented in four groups. The first group, having about 35 definitions, contains all formulae defining the *sets of constituents* as, e.g., a whole document, a *document body*, a *generic part* or a *generic logical description* (see figure 2).

The second group, having also about 35 definitions, contains the formulae defining the *constituents* as, e.g., a *logical object class description*, a *logical object description*, a *content portion description*, a *layout style* or a

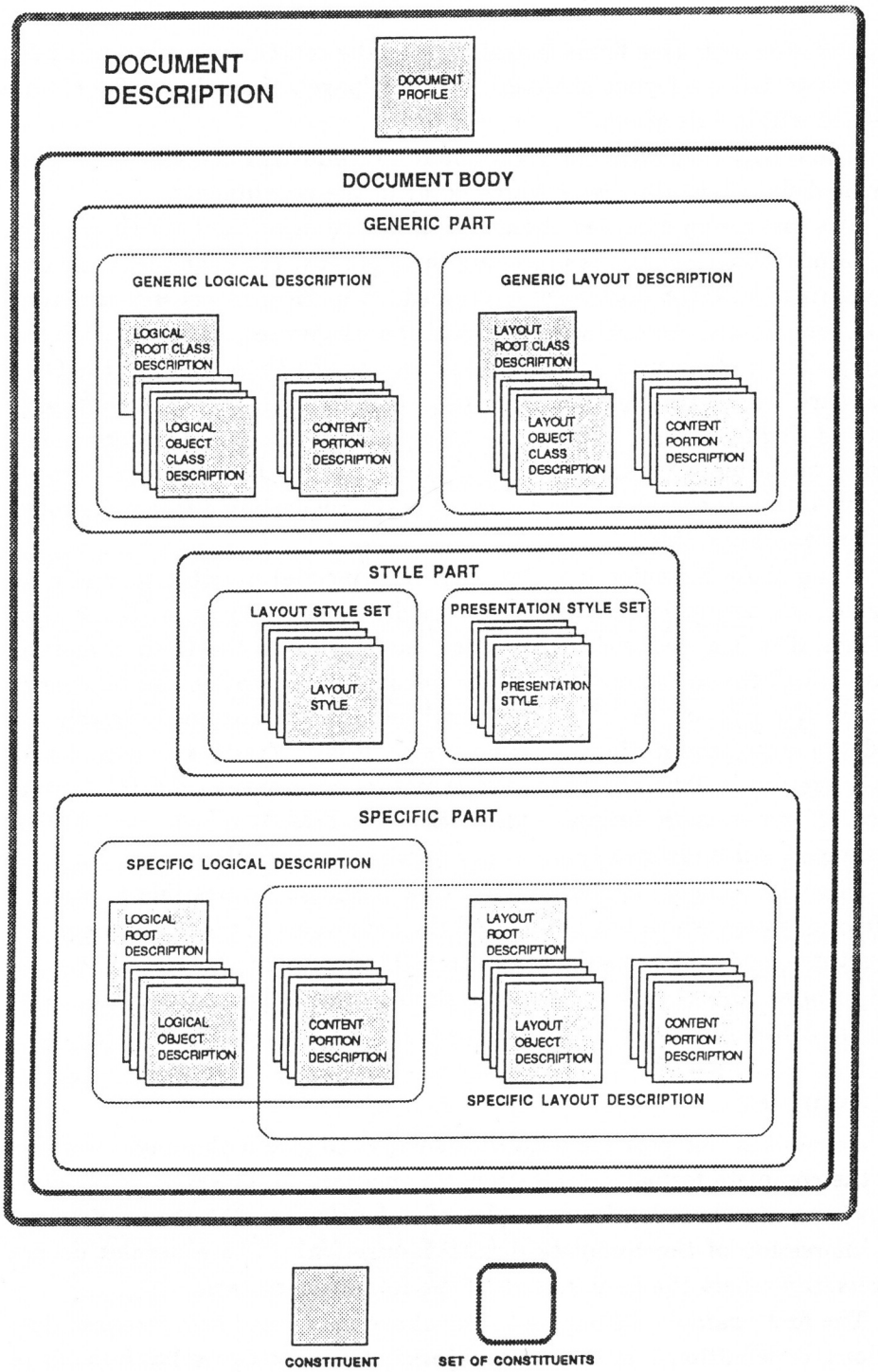

Figure 2: Descriptive representation of a document

presentation style (see figure 2 again). Also the constituents concerning the objects of ODA's layout structure such as 'page', 'frame' and 'block' are defined within this group.

The third group with currently about 40 definitions contains all the formulae defining the *attributes* which occur in the *constituents.*

The last group contains about 35 *subsidiary definitions* which are used in various definitions in the preceding three groups. For example, some constituents of an ODA document are embedded in *hierarchical tree structures* with complicated cross relationships which are rigorously defined within this group. These structures belong, by the way, to the those parts of the ODA Standard which are hard to understand and where a precise definition in natural English is very difficult. By the formal specification a rigorous definition was achieved but, admittedly, this part of the formal specification is a little bit more complex than the other parts, requesting a rather thorough understanding of IMCL.

Using these formulae it is (at least in principle) possible to verify for a given document description, which might be encoded as an ODIF data stream, if it is a document description conforming to the ODA Standard. "Applying" the formulae to a specific document description can be considered as a bundle of "tests" on the constituents of the document description and the cross relationships between them. If each "test" succeeds, i.e., if each formula is True the document structure conforms to the ODA standard. If one or more formulae yield the value False it is not a valid ODA document. An undefined result is not possible for logical reasons.

Though it might not be a trivial task it should be possible to write a software system which has built–in all the definitions of the ODA document structures, and which, reading a given ODA document description, e.g., in the form an ODIF data stream, can decide if that document conforms to the ODA Standard.

7. Examples

Of course it is not possible within this paper to give a thorough overview on the complete formal specification. In this section, however, we want to show some (rather simple) examples of *definitions* so the reader may get an impression of the complete formal description. (The examples do not necessarily reflect the final version of the formal specification.)

The first example, belonging to the above mentioned first group of definitions, is definition 1 in the formal specification (see figure 2). It reads as follows:

Definition 1

1 $\forall doc$
2 $(_0$IsDocument(doc) *iff*
3 $\exists prof$
4 $(_1$IsDocumentProfile$^{35}(prof)$ *and*
5 $(_2 doc = [prof]$ *or*
6 $\exists doby$
7 $(_3$IsDocumentBody$^{2}(doby)$ *and* $doc = [prof] \cup doby$ *and*
8 $(doby)$IsConsistentWith$^{117}(prof)_3)_2)_1)_0)$

The superscripts refer to the numbers of the definitions, where the predicates and operators used within a definition are defined, e.g., the predicate IsDocumentProfile is defined in definition 35. The subscripts attached to some of the parentheses are used to indicate the balancing. These numbers do not belong to the formal description itself but they are, especially in more complicated definitions, to help the readers and, of course, the developers of the formal specification.

The semi–formal explanation of the formula reads as follows:

> Predicate "is a document"
> An entity *doc* is a document or, more precisely, a document description if it is a collection of constituents consisting either of a single document profile *prof* (4, 5) or of a document profile *prof* along with other constituents which constitute a document body *doby* (4, 6, 7). In the latter case the document body has to be consistent with the document profile (8).

The next example, also from the first group of definitions, is definition 2 in the formal specification. It steps down one level and describes the entities which are contained within a document body (see again figure 2). Furthermore, it defines several relations which must hold between those entities. These relationships are not visible in figure 2, but they are described in the natural English specification of the ODA Standard. The definition reads as follows:

Definition 2

1 $\forall doby$
2 $(_0$IsDocumentBody$(doby)$ *iff*
3 $\exists glogct, slogct, glayct, slayct, laystys, prestys$
4 $(_1$IsGenericLogicalStructure$^{22}(glogct)$ *and*
5 IsSpecificLogicalStructure$^{29}(slogct)$ *and*
6 IsGenericLayoutStructure$^{25}(glayct)$ *and*
7 IsSpecificLayoutStructure$^{30}(slayct)$ *and*
8 IsLayoutStyleSet$^{33}(laystys)$ *and*

```
9   IsPresentationStyleSet[34](prestys) and
10  (slogct ⊆ doby or slayct ⊆ doby) and
11  doby ⊆ glogct ∪ slogct ∪ glayct ∪ slayct ∪ laystys ∪ prestys and
12  (laystys ⊆ doby impl glogct ⊆ doby or slogct ⊆ doby) and
13  (slogct ∪ glayct ⊆ doby impl IsRootedGenericLayoutStructure[26](glayct)) and
14  (₂ glogct ⊆ doby and IsRootedGenericLogicalStructure[23](glogct) impl
15    (slogct)IsCompletelyAssociatedWith[31](glogct)₂) and
16  (₃ glayct ⊆ doby and IsRootedGenericLayoutStructure[26](glayct) impl
17    (slayct)IsCompletelyAssociatedWith[31](glayct)₃) and
18  (₄ slogct ∪ slayct ⊆ doby impl
19    (slayct)IsLayoutOfLogicalPartIn[118](doby)₄) and
20  (₅ laystys ∪ glayct ⊆ doby impl
21    ∀a ∈̂ ^laystys • • 'layout object class'
22    (∃g ∈ glayct (C ^g • 'object class identifier' = C a))₅)
23  /* further cross relationships */ ₁)₀)
```

Since the definition is a bit more complicated than in the first example the semi–formal explanation is a bit longer now:

> Predicate "is a document body"
> An entity *doby* is a document body if it is a collection of constituents which includes at least (10) a specific logical structure *slogct* (5) or a specific layout structure *slayct* (7). It may additionally include (11) a generic logical structure *glogct* (4), a generic layout structure *glayct* (6), a layout style set *laystys* (8) and a presentation style set *prestys* (9). A layout style set may be part of the document body only together with a generic or a specific logical structure (12). If a specific logical and a generic layout structure belong to the document body, the generic layout structure is a so–called rooted generic structure (13). If a generic structure is part of the document body and is rooted, the corresponding specific structure is completely associated with the generic structure (14–17). If the document body includes both a specific logical and a specific layout structure, the latter has to be the result of a layout process on the logical part of the document body (18, 19). If a layout style set *laystys* and a generic layout structure *glayct* form part of the document body, all references 'layout object class' from *laystys* to *glayct* are satisfied (20–22)."*

The next example, belonging to the third group of definitions, shows how the formal specification of ODA document structures helped detecting errors or ambiguities in the natural language specification. The draft version of the ODA standard, submitted for voting among the ISO member bodies in

* In the meantime, the term *structure* in the above sense has been replaced by the term *description* (see figure 2).

autumn 1986, contained the following clause which describes the attribute 'separation' used during the layout process of an ODA document. (Parts of the text which are of no relevance here, are omitted and indicated by "...".)

> The attribute 'separation' ... "specifies minimum amounts of separation between the block used to present the content of this logical object and the nearest adjacent block immediately subordinate to the same immediate superior layout object.
>
> Separation may be specified for one or more of:
>
> a) the 'leading edge';
>
> b) the 'trailing edge';
>
> c) the 'centre separation';
>
> The 'leading edge' specifies the minimum separation from the ...
>
> The 'trailing edge' specifies the minimum separation from the ...
>
> The 'centre separation' specifies the minimum distance between ...
>
> The value of this attribute is one or more integers which specify the amount of separation in Basic Measurement Units."

This natural language specification is unclear. For example, if there are three integers specified for the attribute, the standard does not explicitly say, that the first one is associated with the 'leading edge', the second one with the 'trailing edge' and the third one with the 'centre separation'. Although this might be the intuitive interpretation, implementors of the standard might have different views on this point. The standard also allows the specification of less than three integers. But if, e.g., only one integer is specified for the attribute, it could not be determined which edge this integer is associated with.

When trying to formalize this clause of the natural English specification of ODA these difficulties were identified immediately. The resulting definition which was later on also used for the natural English specification, reads as follows:

Definition 85

$$
\begin{array}{ll}
1 & \forall\, v \\
2 & (_0\,\mathrm{IsSeparationValue}(v)\ \underline{\mathit{iff}} \\
3 & (_1\,\mathrm{IsNeNom}^{133}(v)\ \underline{\mathit{and}} \\
4 & \quad \forall\, a\ \hat{\in}\ \hat{}\,v \bullet \\
5 & \quad (_2\,\mathrm{N}\ a \in [\text{'leading edge'}; \text{'trailing edge'}; \text{'center separation'}]\ \underline{\mathit{and}} \\
6 & \quad\ \ \mathrm{IsNumber}(\mathrm{C}\ a)_2)_1)_0)
\end{array}
$$

The predicate IsNeNom(v) (read "is non–empty nomination") used within this definition is True iff v is a nomination and v is not empty, i.e., $v \neq [:]$. The semi–formal explanation of this formula is:

Predicate "is a separation value"
The value of the attribute 'separation' is a non–empty nomination, i.e., a non–empty set of pairs (*name*, *value*). The *name* is an element of the set ['leading edge'; 'trailing edge'; 'center separation']; the *value* is a number.

By using the construct of a *nomination* the ambiguity in the natural language specification was removed.

8. Conclusions

A formal specification technique based on rigorous syntactical and semantical constructs can and should be used to describe information structures when standardizing document processing systems. Especially for complex structures, as in ODA documents, such an approach seems superior to a mere natural language description where, due to the lack of precision in any natural language, ambiguities and even inconsistencies are likely to remain, even when the authors make their best efforts to avoid these.

Although the final form of a standard will still be published in natural English following established regulations of ISO, an accompanying effort for a formal specification will surely improve the standard in various ways and provide a definitive reference document for all questions arising in its interpretation.

9. References

[1] Appelt, W.; Carr, R.; Richter, G. (1987). *Formal Specification of ODA Document Structures.* Draft proposal for an Addendum to IS 8613, Part 2; International Organization for Standardization.

[2] Richter, G; Durchholz, R. (1982). IML–Inscribed High–Level Petri Nets. in: *Information Systems Design Methodologies*; Ed.: Olle, T. W.; Sol, H. G.; Verrijn–Stuart; North–Holland.

[3] Richter, G; Durchholz, R. (1988). *Information Modelling by Composition — The IMC/IMCL Reference Manual.* Technical Report, GMD, Sankt Augustin.

[4] ISO8613. *Office Document Architecture (ODA) and Interchange Format* (1988). International Standard IS 8613, International Organization for Standardization.